Système et procédé de contrôle d'équipements à distance à l'aide de fonctions API, dispositif et module de radiocommunication et jeu de fonctions correspondants.

Le domaine de l'invention est celui du contrôle à distance d'équipements, et notamment d'équipements limités en ressources de traitement de données. Ainsi, l'invention s'applique par exemple aux systèmes de relevé de données à distance, par exemple sur des compteurs d'eau, de gaz ou d'électricité, et plus généralement aux systèmes de télémétrie, de suivi de commandes, et plus généralement de contrôle de machines (en anglais « M to M : machine to machine »).

Il existe déjà diverses solutions pour réaliser de telles opérations. Elles ont généralement été développées de façon spécifique pour une application donnée. En d'autres termes, il s'agit de solutions « propriétaires », qui sont difficilement adaptables à d'autres applications.

On connaît par ailleurs un protocole, développé par les sociétés IBM et ARCOM Control Systems (marques déposées), connu sous le nom de technologie « MQIsdp Messaging ». Cette technique propose un protocole de communication entre un ou plusieurs équipements limités en ressources, et un ou plusieurs serveurs (« brokers » en anglais), en utilisant un lien TCP/IP.

Cependant, même avec ce protocole spécifique, il est nécessaire d'adjoindre aux équipements des moyens de traitement spécifiques (microprocesseurs, mémoires, ...), qui permettent d'instaurer le dialogue avec ces serveurs distants, selon le format MQIsdp requis. La liaison entre l'équipement et le serveur peut utiliser une liaison de type téléphonique, à l'aide d'un modem.

Dans de nombreuses applications, il serait cependant souhaitable de pouvoir se passer d'une liaison téléphonique filaire. On peut alors envisager la mise en œuvre de moyens de radiocommunication, par exemple selon la norme GSM ou GPRS.

Dans ce cas, on utilisera un équipement de radiotéléphonie pour assurer la fonction de modem. Cependant, il reste nécessaire, selon l'art antérieur, d'associer à l'équipement des moyens spécifiques et propriétaires de traitement de données pour établir et réaliser l'échange de données avec le serveur.

Cet aspect est une limitation très importante au développement des applications mentionnées ci-dessus, et de nombreuses autres applications que permet d'envisager le protocole MQIsdp.

L'invention a notamment pour objectif de pallier cet inconvénient de l'art antérieur.

Il convient de noter que le fait d'identifier ce problème est en soi une partie de l'invention. En effet, l'homme du métier est persuadé qu'il est absolument nécessaire d'équiper les équipements terminaux de moyens de traitement suffisants, et ne peut en aucun cas envisager qu'il est possible de les réduire, voire de les supprimer.

C'est pourtant un objectif de l'invention que de permettre de simplifier les traitements nécessaires du côté des équipements, et d'éviter que ceux-ci doivent disposer de moyens complexes et coûteux tels qu'un micro-processeur.

Un autre objectif de l'invention est de proposer une technique simple et générique, permettant d'instaurer facilement et efficacement un dialogue avec un serveur selon le protocole MQIsdp.

Ainsi, un objectif particulier de l'invention est de permettre d'assurer un tel dialogue de façon simple et peu coûteuse, à l'aide d'un simple module de radiocommunication.

Encore un autre objectif de l'invention est de fournir une telle technique, permettant d'établir une liaison entre des serveurs et des équipements par voie radiotéléphonique, de façon simple, standardisée et peu coûteuse.

L'invention a également pour objectif de fournir une telle technique, permettant de développer un nombre important d'applications, sans qu'il soit nécessaire de développer des applications spécifiques à chaque fois.

Un autre objectif de l'invention est de fournir une telle technique, ne

nécessitant pas une connaissance du protocole MQIsdp dans les applications développées.

Encore un autre objectif de l'invention est de fournir une telle technique, qui est à la fois techniquement simple et évolutive, et adaptable à diverses situations (par exemple pour la taille des données à échanger) et aux éventuelles évolutions futures.

Ces objectifs, ainsi que d'autres qui apparaîtront plus clairement par la suite, sont atteints selon l'invention à l'aide d'un système de contrôle d'équipements à distance, permettant l'interconnexion entre au moins un serveur et au moins un équipement distant selon le protocole MQIsdp.

Selon l'invention, le système de contrôle associe à au moins un desdits équipements distants des moyens de radiocommunication capable de traiter en interne un protocole de communication mettant en oeuvre des fonctions source de type API disponibles dans une plateforme logicielle (Open AT) permettant d'embarquer au moins une application, et lesdits moyens de radiocommunication sont dotés d'un jeu de fonctions (API) spécifiques permettant d'échanger des données avec au moins un serveur mettant en œuvre ledit protocole MQIsdp, de façon à permettre une interconnexion entre le ou lesdits serveurs et le ou lesdits équipements distants via lesdits moyens de radiocommunication, ces derniers gérant également au moins une application entre le ou lesdits serveurs et le ou lesdits équipements distants.

Ainsi, il est possible de gérer entièrement en interne, dans les moyens de radiocommunication, et en particulier dans un module, l'application de contrôle d'un ou plusieurs terminaux avec un serveur fonctionnant selon le protocole MQIsdp, sans que le terminal ne connaisse ce protocole. Il n'y a rien à rajouter, du côté du terminal (ni matériel, tel que microprocesseur ou mémoire, ni logiciel, tel qu'une application dédiée). C'est le module qui gère ces opérations, à l'aide des fonctions de l'invention, et assure l'interface avec le protocole MQIsdp.

Selon un mode de réalisation avantageux, lesdits moyens de radiocommunication comprennent un module de radiocommunication, regroupant

sur un même substrat l'ensemble des moyens de traitement de données radio-fréquence et bande de base, ainsi que les moyens de gestion desdites fonctions (API) et la ou lesdites applications. C'est ce module qui intègre l'application et les fonctions source (API).

De façon préférentielle, lesdits moyens de radiocommunication intègrent ledit protocole MQIsdp sous la forme d'une librairie, définissant ledit jeu de fonctions (API) spécifiques.

Au moins dans un premier mode de mise en oeuvre, lesdits moyens de radiocommunication gèrent uniquement la signalisation d'un échange de données, lesdites données étant transférées directement d'un équipement distant vers un serveur, ou inversement.

Dans un second mode de mise en œuvre avantageux, qui est celui actuellement développé, lesdits moyens de radiocommunication gèrent la signalisation d'un échange de données et le transfert desdites données, ces dernières étant temporairement stockées dans au moins une mémoire tampon.

Dans ce cas, la taille de la ou desdites mémoires tampon est avantageusement paramétrable.

Si les deux modes de réalisation sont disponibles, on prévoit avantageusement que l'on fonctionne dans ledit premier mode lorsque la taille de la ou desdites mémoires tampon vaut 0, et dans ledit second mode sinon.

De façon avantageuse, ledit jeu de fonctions API spécifiques comprend des fonctions permettant :

- la connexion à un desdits serveurs ;
- l'envoi de messages ;
- la réception de messages ;
- configuration d'au moins un paramètre.

Préférentiellement, au moins certaines desdites fonctions (API) spécifiques sont organisées de façon à pouvoir assurer au moins deux opérations et/ou agir sur au moins deux aspects distincts, en fonction d'un paramétrage prédéfini.

Cela permet d'optimiser le nombre de fonctions, tout en rendant possible

des évolutions.

Dans un mode de réalisation préférentiel, ledit jeu de fonctions (API) comprend uniquement 12 fonctions.

De façon avantageuse, ledit jeu de fonctions (API) spécifiques comprend une fonction d'initialisation (mqisdp_init) rétablissant des paramètres par défaut, et devant être appelée au moins une fois avant l'utilisation d'autres fonctions (API).

Préférentiellement, ledit jeu de fonctions (API) spécifiques comprend une fonction (mqisdp_resume) appelée lorsque une liaison IP a été établie.

Avantageusement, il comprend une fonction d'établissement d'une connexion avec un desdits serveurs (mqisdp_connect), permettant de définir des paramètres de ladite connexion, et une fonction de déconnexion (mqisdp_disconnect) de ladite connexion.

Ladite fonction d'établissement d'une connexion permet notamment, avantageusement, de sélectionner un mode de transmission parmi au moins deux (GSM et GPRS).

Ledit jeu de fonctions comprend encore, avantageusement, une fonction (mqisdp_publish) pour l'envoi d'un message vers un desdits serveurs.

Préférentiellement, il comprend une fonction d'abonnement à un desdits serveurs (mqisdp_subscribe), et une fonction de désabonnement (mqisdp_unsubscribe) audit serveur.

De façon avantageuse, ledit jeu de fonctions comprend au moins une fonction de demande d'information sur au moins un aspect d'une communication en cours, et par exemple au moins une des fonctions appartenant au groupe comprenant :

- une fonction de demande du statut d'une connexion (mqisdp_getConStatus) ;
- une fonction de demande du statut d'un message donné (mqisdp_getMsgStatus) ;
- une fonction de demande de la taille actuelle d'une file d'attente

(mqisdp_getQueueSize) ;

- une fonction de demande de l'espace disponible dans une file d'attente (mqisdp_getAvailableSize).

Préférentiellement, il comprend une fonction de définition de la taille d'une file d'attente (mqisdp_setQueueSize).

L'invention concerne également les procédés de contrôle d'équipements à distance, permettant l'interconnexion entre au moins un serveur et au moins un équipement distant selon le protocole MQIsdp.

Un tel procédé associe à au moins un desdits équipements distants des moyens de radiocommunication capable de traiter en interne un protocole de communication mettant en oeuvre des fonctions source de type API disponibles dans une plateforme logicielle (Open AT) permettant d'embarquer au moins une application, et met en œuvre, dans lesdits moyens de radiocommunication, un jeu de fonctions API spécifiques permettant d'échanger des données avec au moins un serveur mettant en œuvre ledit protocole MQIsdp, de façon à permettre une interconnexion entre le ou lesdits serveurs et le ou lesdits équipements distants via lesdits moyens de radiocommunication, ces derniers gérant également au moins une application entre le ou lesdits serveurs et le ou lesdits équipements distants.

L'invention concerne encore les dispositifs et les modules de radiocommunication mis en œuvre dans un système de contrôle d'équipements à distance stel que décrit ci-dessus.

L'invention concerne enfin également un jeu de fonctions (API) mis en oeuvre dans un système de contrôle d'équipements à distance, permettant d'échanger des données avec au moins un serveur mettant en œuvre ledit protocole MQIsdp.

D'autres caractéristiques et avantages de l'invention apparaîtront plus clairement à la lecture de la description suivante d'un mode de réalisation préférentiel de l'invention, donné à titre de simple exemple illustratif et non limitatif, et des dessins annexés, parmi lesquels :

- la figure 1 illustre un exemple de système dans lequel l'invention peut

être mise en œuvre ;

- la figure 2 est un exemple d'intégration du protocole MQIsdp dans un module selon l'invention ;
- la figure 3 est un synoptique simplifié d'un exemple d'envoi de message mettant en œuvre l'invention.

1. Rappels sur le protocole MQIsdp (marque déposée)

Le protocole MQIsdp (« WebSphere MQ Integrator SCADA device protocol » en anglais) est une norme ouverte développée par IBM et Arcom Control Systems (marques déposées), visant à permettre les échanges de données (sous forme de messages) depuis des dispositifs distants (ou terminaux), généralement de bas de gamme et disposant de peu de puissance de traitement, vers un serveur, appelé également par la suite « courtier » (« broker » en anglais) WebSphere MQ Integrator par TCP/IP, et inversement.

MQIsdp (également appelé par la suite Wavecom SCADA) est un protocole de transfert de données (message) basé sur un modèle de communication du type publier/souscrire, disponible libre de droit sur Internet. Il peut être décrit comme une simple couche de gestion de données agnostiques au-dessus du protocole TCP/IP pour assurer la gestion et les accusés de réception de message requis afin d'assurer la livraison fiable du message.

Dans le modèle de communication publier/souscrire, les données sont échangées entre un producteur/consommateur de données (le client) et un courtier de message (le serveur). Le courtier de messages peut être considéré comme un « hub » (nœud) de commutation multiprotocole au niveau du protocole de l'application qui reçoit des messages, les transforme, les reformate, etc. en d'autres structures en fonction d'un modèle de données défini par l'utilisateur.

Enfin, les messages éventuellement transformés peuvent être envoyés par le courtier (publiés) aux clients (dispositif de zone, ERP, SAP, Oracle, SQL, etc.) abonnés en utilisant les cartes clients appropriées. Le courtier peut, évidemment, publier également des messages ne provenant pas d'un client.

Le courtier de messages gère tous les messages entrant et sortant d'une

rubrique. Un client publie des messages dans/avec une rubrique ou souscrit à des messages de/par une rubrique identifiant le flux de message du courtier de message vers lequel ou à partir duquel le message doit être publié.

La spécification MQIsdp définit un ensemble de messages très simple, comprenant : « connect » (connecter), « disconnect » (déconnecter), « publish » (publier), « subscribe » (abonner), « unsubscribe » (désabonner).

2. principes de l'invention

2.1 généralités

L'invention concerne donc une nouvelle approche du contrôle d'équipements à distance, reposant notamment sur la mise en œuvre d'un jeu de fonctions spécifiques API permettant à une application externe de gérer des échanges de données entre un terminal distant et un serveur, via des moyens de radiocommunication (par exemple un module de type Wismo (marque déposée)), sans que l'application connaisse le protocole MQIsdp mis en œuvre par le serveur. Ce sont les moyens de radiocommunication qui gèrent cet aspect, et par exemple les acquittements prévus dans le protocole MQIsdp.

Les fonctions API sont des fonctions développées en langage C, qui permettent au module de gérer en interne, sous le contrôle d'une application elle-même interne au module, des échanges de données avec des serveurs mettant en oeuvre le protocole MQIsdp.

La figure 1 illustre de façon simplifiée le principe de l'invention. L'objectif est de faire communiquer tout type de machines distantes, par exemples des instruments de mesure 11 avec une ou plusieurs applications hébergées par des serveurs 12, capables de recevoir des données 13 selon le protocole MQIsdp, et de les transformer, traiter ou transmettre.

Selon l'invention, on associe aux terminaux (ou machines) distants 11 des moyens de radiocommunication 14, par exemple sous la forme d'un module Wismo (marque déposée), embarquant notamment les outils de développement distribués par le déposant sous la marque « Muse platform ».

2.2 Notion de module

Pour mémoire, on rappelle que la plupart des dispositifs de radiocommunication comprennent, de façon classique, un ensemble de composants électroniques implantés sur un circuit imprimé. Ces différents composants ont pour but d'assurer les différentes fonctions nécessaires, depuis la réception d'un signal RF jusqu'à la génération d'un signal audible (dans le cas d'un radio-téléphone), et inversement. Certaines de ces fonctions sont analogiques, et d'autres numériques.

La fabrication de ces dispositifs de radiocommunication est un sujet de recherche important. En effet, on vise au moins trois objectifs difficiles à concilier : miniaturiser les dispositifs, augmenter les fonctionnalités et simplifier le montage. On sait notamment que l'implantation des différents composants sur le circuit imprimé est une opération relativement complexe, de nombreux composants devant être mis en place sur une surface de plus en plus restreinte, du fait des exigences de miniaturisation.

La conception de ces systèmes est donc complexe, puisqu'elle nécessite en outre d'associer des composants divers, souvent de sources multiples, qu'il faut faire fonctionner ensemble, en respectant les spécificités de chacun. Par ailleurs, après le montage de l'ensemble des composants, des phases de calibration et de tests, souvent longues et complexes, sont nécessaires pour garantir le bon fonctionnement du dispositif.

Enfin, malgré la réduction de la taille de certains composants, l'ensemble occupe une certaine surface, qu'il est difficile de réduire.

Le titulaire de la présente demande de brevet a proposé une approche palliant un certain nombre de ces inconvénients, consistant à regrouper dans un module unique, toutes ou au moins la plupart, des fonctions d'un dispositif de radiocommunication numérique.

Un tel module se présente sous la forme d'un boîtier unique et compact, préférentiellement blindé, que les fabricants de dispositifs peuvent implanter directement, sans devoir prendre en compte une multitude de composants.

Ce module (encore appelé parfois « macro-composant ») est en effet formé d'un regroupement de plusieurs composants sur un substrat, de façon à être implanté sous la forme d'un unique élément. Il comprend les composants et les logiciels essentiels nécessaires au fonctionnement d'un terminal de télécommunication utilisant des fréquences radio-électriques. Il n'y a donc plus d'étapes complexes de conception du design, et de validation de celui-ci. Il suffit de réserver la place nécessaire au module.

Un tel module permet donc d'intégrer facilement, rapidement et de façon optimisée l'ensemble des composants dans des terminaux sans-fil (téléphones portables, modems, ou tout autre application exploitant un standard sans fil).

Par ailleurs, celui-ci regroupant toutes les fonctions essentielles et ayant été conçues comme un tout, les problèmes de calibration et de tests ne se posent plus de la même manière, ou sont à tout le moins, grandement simplifiés.

Ainsi, les modules diffusés par le titulaire de la présente demande de brevet sont entièrement testés tant sur le plan matériel (« hardware ») que logiciel (« software ») sur la plupart des réseaux sur lesquels ils pourront être utilisés ensuite. En outre, le module englobe avantageusement les aspects de propriété industrielle (toutes les fonctions ayant été regroupées, c'est le fabricant du module qui gère les aspects de droits de propriété industrielle correspondants) et d'assistance technique.

2.3 fonctions API

On connaît déjà des modules embarquant des fonctions API. Le document de brevet FR-0103909, présente ainsi un module de radiocommunication, du type hébergeant et exécutant un logiciel principal assurant notamment des fonctions de radiocommunication, ledit logiciel principal comprenant des moyens d'exécution de commandes de pilotage, envoyées au logiciel principal par au moins un logiciel client de pilotage et appartenant à un jeu de commandes de pilotage prédéterminé.

Selon cette technique, le module de radiocommunication héberge et exécute en outre au moins un logiciel client, dit logiciel client embarqué. En outre, le logiciel client embarqué et le logiciel principal comprennent des moyens

permettant au logiciel client embarqué de jouer au moins un des deux rôles suivants :

- le rôle d'un logiciel client de pilotage, envoyant des commandes de pilotage au logiciel principal, et recevant du logiciel principal des réponses, résultant de l'exécution de certaines des commandes de pilotage ;
- le rôle d'un logiciel client de supervision, gérant l'exécution de commandes de pilotage envoyées par un logiciel client de pilotage, dit logiciel client externe, hébergé et exécuté par un équipement tiers coopérant avec le module de radiocommunication.

Le principe général de l'invention consiste donc à héberger sur le module de radiocommunication au moins un logiciel client pouvant jouer le rôle d'un logiciel client de pilotage et/ou le rôle d'un logiciel client de supervision.

On pourra se référer à ce document pour plus de détails, si nécessaire.

2.4 Nouvelles fonctions API

Le module 14 est donc capable, selon l'invention, de gérer des fonctions API, et en nombre réduit, permettant un dialogue simple et efficace avec un terminal, sous le contrôle d'une application interne. Il assure la transformation au format MQIsdp, et gère l'émission et la réception de données 15 selon ce protocole, de façon transparente pour l'application et le terminal.

L'échange de données peut ainsi se faire de façon hertzienne 16, par exemple selon le standard GSM ou GPRS. Vu du serveur 12, les informations sont au format MQIsdp. Pour les terminaux 11, il n'est pas nécessaire de connaître ce protocole, ni de mettre en œuvre des moyens particuliers, tels qu'un microprocesseur et des mémoires, et une application dédiée.

Comme on le verra par la suite, les fonctions proposées peuvent être en nombre limité, tout en permettant des évolutions.

Les données transitent par le module 14. Elles sont alors stockées temporairement dans des buffers (mémoires tampon), dont la taille est

configurable en fonction des besoins.

La figure 2 illustre un exemple simplifié d'architecture logicielle pouvant être mise en œuvre dans le module 14.

Un tel module 14 comprend généralement :

- une couche logicielle de base 21 (« Wavecom Core SoftWare ») ;
- une bibliothèque Open AT 22 (« Open AT Library ») ;
- une bibliothèque ADL 23 (« ADL Library ») ;
- une bibliothèque TCP/IP 24 (« TCP/IP Library ») ;
- une couche applicative 25 (« Open AT Application »).

Selon l'invention, on prévoit donc en outre une bibliothèque 26 de commandes spécifiques (« Wavecom SCADA Protocol Library ») pour communiquer selon le protocole MQIsdp, qui se place au dessus de la bibliothèque TCP/IP 24.

L'interface proposée comprend, dans cette bibliothèque 26, seulement 12 fonctions API, permettant d'exploiter entièrement le protocole MQIsdp. Ces fonctions sont détaillées par la suite.

Optionnellement, les commandes AT 27 s'adressent, selon les cas, à la couche de base 21, à la bibliothèque TCP/IP 24 ou à la bibliothèque SCADA 26. Le module peut en effetégalement gérer des commandes AT, pour assurer les mêmes opérations d'interface.

3. description détaillée d'un mode de réalisation de fonctions API

On décrit ci-après les fonctions API pouvant être utilisées pour piloter le protocole Wavecom SCADA 26.

3.1 Documents connexes

En cas de besoin, on pourra se référer à :

[1] la fiche "WepSphere MQ Integrator SCADA Device Protocol" figurant à l'annexe B du manuel de référence "IBM WebSphere MQ Integrator Programming Guide" disponible à l'adresse suivante : http://publifp.boulder.ibm.com/epubs/odf/bipval04.pdf

[2] Wavecom AT Commands Interface Guide

Référence : WM_SW_OAT_IFS_001 - révision : 009 ou suivantes.

Ce document décrit les commandes AT prises en charge par le produit Wavecom permettant de gérer les événements ou services GSM connexes.

[3] AT Command Interface for TCP/IP

Révision : 1.7

Ce document décrit les paramètres et le jeu de commandes AT permettant la configuration et le pilotage de la superposition TCP/IP et des protocoles disponibles sur les produits Wavecom.

[4] WebSphere MQ Integrator Programming Guide (Version 2.1), Annexe B.

[5] Edjlb-30x Interface Specification

[6] ADL User Guide

3.2 Abréviations et définitions

3.2.1. Abréviations

MQIsdp	Protocole de dispositif MQ Integrator SCADA
SCADA	Supervisory Control and Data Acquisition (Dispositif de surveillance et acquisition des données)
APN	Access Point Name (Nom de point d'accès)
AT	Attention
DNS	Domain Name Système (Système de noms de domaine)
ISP	Internet Service Provider (Fournisseur de services sur Internet)
ME	Mobile Equipment (Matériel mobile)
MS	Mobile station (Station mobile)
QoS	Quality of Service (Qualité de service)

3.3 architecture

La bibliothèque MQIsdp est construite au-dessus de la bibliothèque ADL de Wavecom et de la bibliothèque TCP/IP. Des applications imbriquées utilisant la bibliothèque MQIsdp doivent donc être mises en œuvre à l'aide de la bibliothèque ADL (plutôt qu'avec la couche API Open AT standard directement).

La figure 2 déjà commentée présente cette architecture.

Le logiciel « MQIsdp for Open AT » selon ce mode de réalisation contient les éléments suivants :

- Une bibliothèque logicielle conforme à ADL (wmmqisdp.lib)
- Un fichier d'en-tête (mqisdp.h) définissant l'API MQIsdp
- Quelques échantillons de code source

Une application imbriquée utilisant la bibliothèque MQIsdp doit être reliée à wmadi.lib et edlib.lib (ces bibliothèques sont incluses dans le logiciel « MQIsdp for Open AT » fourni avec le module).

3.4 Remarques relatives à la gestion de mémoire

En raison des ressources de mémoire limitées dans l'environnement Open AT, il est important de connaître exactement la quantité de mémoire utilisée par une bibliothèque ou un processus donné. La bibliothèque MQIsdp est donc centrée sur une gestion précise de la mémoire.

Toutes les fonctions de commande MQIsdp dans l'application API (connect (connecter), publish (publier), subscribe (abonner), unsubscribe (désabonner) et disconnect (déconnecter)) sont asynchrones (les fonctions renvoient immédiatement et la sortie finale des opérations associées est signalée par rappel).

Il peut exister à tout moment des messages MQIsdp que la bibliothèque a accepté, mais qui n'ont pas été envoyés au courtier, qui n'ont pas encore reçu d'accusés de réception de la part du courtier ou qui n'ont pas encore été distribués à l'application imbriquée (le client). Tous ces messages en suspens sont conservés dans la file d'attente par la bibliothèque. La bibliothèque ne peut pas garantir la taille maximale de cette file d'attente car elle dépend de la taille des données à envoyer/recevoir, de la fréquence d'envoi/réception et de la largeur de bande de la connexion TCP/IP.

Un contrôle complet de la taille maximale de la file d'attente est fourni à l'application client par un ensemble de fonctions API permettant de définir et de connaître la taille maximale de la file d'attente et de demander la taille actuelle de

la file d'attente.

Dans certains cas (taux élevé d'erreurs de transmission entraînant des coupures fréquentes de la connexion), une taille plus élevée de la file d'attente est nécessaire car la bibliothèque doit mettre les messages en mémoire tampon jusqu'à ce que la communication soit rétablie.

3.5 remarques complémentaires

3.5.1 Une seule connexion MQIsdp à la fois

La bibliothèque MQIsdp dépend de la bibliothèque TCP/IP, qui est limitée à une connexion à la fois. La bibliothèque MQIsdp est donc limitée à une seule connexion MQIsdp à la fois.

3.5.2 Taille de message MQIsdp limitée

La bibliothèque peut être conçue pour envoyer et recevoir des messages MQIsdp avec une taille maximale de 65 535 octets (y compris des en-têtes fixes et variables).

La spécification MQIsdp autorise des longueurs de charge utile maximales de 268 435 455 octets, ce qui signifie qu'un courtier peut potentiellement envoyer un message supérieur à 65 535 octets (ou la taille limite de la file d'attente) au client. Dans ce cas, le client est prévenu par un programme de traitement que la file d'attente est saturée (et le message n'est pas reçu). De même, si le client essaie d'envoyer des messages alors que la file d'attente est saturée, une indication signale que la file d'attente est saturée et que l'envoi du message n'a pas été accepté.

Il est ainsi aisé de gérer la saturation éventuelle d'une file d'attente et/ou la transmission d'un message de taille trop importante.

Comme déjà mentionné, on peut également, dans certains modes de réalisation, prévoir des transferts directs entre un terminal et le serveur, le module ne gérant que la signalisation.

3.6 API MQIsdp

Les sections suivantes donnent une description détaillée de toutes les

fonctions de l'API MQIsdp.

3.6.1 En-tête requis

L'en-tête des fonctions MQIsdp est : Mqisdp.h

3.6.2 Fonction mqisdp_init

Cette fonction DOIT être appelée pour initialiser la bibliothèque MQIsdp ; elle doit être appelée au moins une fois avant l'utilisation des autres fonctions API.

La fonction rétablit (ou réinitialise) les valeurs par défaut des paramètres de connexion et des options de connexion. Elle règle la taille de la file d'attente sur 3 Ko. Si la fonction est appelée lorsqu'une connexion est active, elle déconnecte brutalement et supprime tous les messages en file d'attente.

La bibliothèque TCP/IP DOIT être initialisée avant l'initialisation de la bibliothèque MQIsdp.

a - Prototype

```
u8 mqisdp_init (mqisdp_linkHdlr_f LinkHandler);
```

b - Paramètres

* LinkHandler :

Le paramètre LinkHandler est appelé par la bibliothèque lorsqu'une liaison IP est nécessaire mais absente ou lorsque la liaison IP est présente mais pas nécessaire. Il permet donc à l'application client de contrôler précisément la liaison IP.

Le type suivant est utilisé pour le paramètre LinkHandler :

typedef void (*mqisdp_linkHdlr_f)(u8 Event) ;

Le paramètre Event peut être :

- MQISDP_LINK_NEEDED si une liaison IP est nécessaire, mais qu'aucune liaison n'est établie.
- MQISDP_LINK_CLOSABLE si une liaison IP n'est plus nécessaire mais qu'une liaison est encore établie.

L'appelant peut soit fournir un paramètre LinkHandler conforme au prototype ci-dessous, soit utiliser NULL. Si le paramètre LinkHandler est NULL,

la bibliothèque assurera le contrôle de la liaison IP et essaiera de (ré)établir une liaison IP lorsqu'l est nécessaire et de fermer la liaison lorsqu'elle n'est plus nécessaire. Dan ce cas, la composition, les paramètres PPP et/ou CPRS doivent avoir été activés par l'API TCP/IP avant la connexion à un courtier de message.

c - Valeurs de retour

MQISDP_OK : L'initialisation (ou la réinitialisation) a abouti.

3.6.3 - Fonction mqisdp_resume

Cette fonction doit être appelée lorsqu'une liaison IP a été établie après que la bibliothèque MQIsdp a indiqué qu'une liaison IP était nécessaire (MQISDP_LINK_NEEDED) par l'intermédiaire du programme de traitement de liaison.

a - Prototype

```
void mqisdp_resume ();
```

b - Paramètres

Aucun.

c - Valeurs de retour

Aucune.

3.6.4 - Fonction mqisdp_connect

Cette fonction est utilisée pour établir une connexion avec un courtier de message. La fonction est asynchrone et le résultat final de l'opération est signalé par une fonction de rappel (ConCtrlHandler). Si une connexion est déjà active lorsque cette fonction est appelée, une erreur est renvoyée.

a - Prototype

```
u8 mqisdp_connect (
mqisdp_connectionParams_t* ConnectionParams,
mqisdp_connectionOptions_t ConnectionOptions,
mqisdp_connectionWill_t ConnectionWill,
mqisdp_conCtrlHdlr_f ConCtrlHandler,
mqisdp_msgCtrlHdlr_f MsgCtrlHandler,
mqisdp_subCtrlHdlr_f SubCtrlHandler,
```

```
mqisdp_msgHdlr_f MsgHandler,
);
```

b - Paramètres

* ConnectionParams : Pointeur vers une structure contenant les informations de connexion de base. Le pointeur ne doit PAS être avoir une valeur nulle ; la structure utilise le type suivant :

```
typedef struct
{
// Client id with a maximum length of 29
ascii*Clientid;
//Address of message broker (ip or name, according to the settings of the
TCP/IP stack)
ascii*BrokerAddress;
}
    mqisdp_connectionParams_t;
```

* ConnectionOptions :

Pointeur vers une structure contenant les informations de connexion facultatives utilisant le type suivant :

```
typedef struct
{
//Port on which to connect to the message broker
u16 Port;//default : 1883
//Keep Alive Timer; C=no keep-alive messages
u16 KeepAliveTimer; // default :0
//Retry count; how many times a connection attempt or message sending
should be retried
u16 RetryCount://default : 10
//Retry delay : how long the library should at least wait before retrying (in
seconds)
```

u16 RetrayDelay;//default : 10

//Clean flag : indicates whether each a client connection should be performed or not!

bool nCleanConnection://default : TRUE

}

mqisdp_connectionOptions_t;

Le pointeur peut avoir la valeur nulle (NULL). Dans ce cas, les valeurs par défaut sont utilisées.

* ConnectionWill :

Pointeur vers une structure contenant les informations sur le will de connexion. La structure utilise le type suivant :

typedef struct

{

//Quality of service

u8 Qos;

bool Retain;

ascii* Topic;

u18 PayloadLength;

u8 Payload [3];

}

mqisdp_connectionwill_t;

Le pointeur peut avoir la valeur nulle (NULL). Dans ce cas les valeurs par défaut sont utilisées.

* ConCtrlHandler :

Programme de traitement de contrôle de connexion par lequel le client reçoit des événements relatifs à la connexion. Le programme de traitement utilise le type suivant :

typedef void (*mqisdp_conCtrlHdlr_f) (u8 Event, u8 ErrorCode);

Le paramètre Event peut avoir les valeurs suivantes :

MQISDP_CON_OPEN : La connexion au courtier est prête.

MQISDP_CON_ERR : La connexion n'a pas pu être établie en raison d'une erreur ou s'est interrompue (la bibliothèque peut tenter de la rétablir).

MQISDP_CON_ACCEPTED : La connexion mqisdp avec le courtier est prête.

MQISDP_CON_REFUSED : La connexion mqisdp avec le courtier a été refusée.

MQISDP_CON_IN_Q_FULL : Des données entrantes ont été refusées car la file d'attente est saturée.

MQISDP_CON_BROKEN : La connexion mqisdp avec le courtier s'est interrompue et toutes les tentatives de reconnexion ont échoué.

MQISDP_CON_CLOSED : La connexion avec le courtier a été fermée (proprement).

MQISDP_CON_DISCONNECTED : La connexion mqisdp avec le courtier a été fermée (proprement).

Dans un scénario normal (depuis la connexion jusqu'à la déconnexion), le programme de traitement de contrôle reçoit la séquence suivante d'événements :

MQISDP_CON_OPEN

MQISDP_CON_ACCEPTED

MQISDP_CON_CLOSED

MQISDP_CON_DISCONNECTED

L'envoi de messages n'est possible que lorsque la connexion MQIsdp est prête (c'est-à-dire lorsque la fonction MQISDP_CON_ACCEPTED a été reçue et qu'aucune déconnexion n'a eu lieu ou n'a été effectuée.

Le paramètre Errorcode n'est utilisé que si le paramètre Event est MQISDP_CON_REFUSED. Dans ce cas, le code d'erreur peut prendre les formes suivantes :

MQISDP_ERR_PROTOCOL_VERSION_ERROR : Connexion refusée en raison d'une version de protocole non prise

en charge.

MQISDP_ERR_INDENTIFIER_REJECTED : Connexion refusée en raison d'un rejet de l'identificateur.

MQISDP_ERR_BROKER_UNAVAILABLE : Connexion refusée en raison de l'indisponibilité du courtier.

MQISDP_TCPIP_UNAVAILABLE : La connexion n'a pas pu être établie en raison d'une erreur sur la liaison TCP/IP.

MQISDP_DNS_PROBLEM : La connexion n'a pas pu être établie en raison d'une erreur SND.

MQISDP_ERR_SOCKET_ERROR : La connexion n'a pas pu être établie en raison d'une erreur de socket.

* MsgCtrlHandler :

Programme de traitement de contrôle des messages par lequel le client reçoit des événements relatifs à l'envoi du message. Le programme de traitement utilise le type suivant :

typedef void (*mqisdp_msgCtrlHdlr_f) (u8 Event, u16 Msgid);

Le paramètre Event peut prendre les formes suivantes :

MQISDP_MSG_DELIVERED : Le message a été envoyé/distribué (conformément au paramètre Qos).

MQISDP_MSG_DISCARDED : Le message a été supprimé (échec de toutes les tentatives).

Le paramètre Msgid contient l'id du message concerné par l'événement. L'id d'un message est renvoyé lors de l'utilisation de l'une des fonctions suivantes : mqisdp_publish, mqisdp_subscribe, mqisdp_unsubscribe, mqisdp_disconnect.

* SubCtrlHandler :

Programme de traitement de contrôle d'abonnement par lequel (s'il n'a pas la valeur nulle (NULL)), le client reçoit des événements relatifs à l'accusé de réception de l'abonnement. Si le programme de traitement a la valeur NULL, les événements d'accusé de réception de l'abonnement sont transférés au programme de traitement de contrôle des messages.

En revanche, les informations relatives au niveau accordé de qualité de service pour chacune des rubriques souscrites ne peuvent pas être reçues par l'intermédiaire du programme de traitement de contrôle des messages.

Le programme de traitement de contrôle d'abonnement utilise le type suivant :

```
typedef void (*mqisdp_subCtrlHdlr_f) (
u8 Event,
mqisdp_subscriptionParams_t* SubscriptionAray
);
```

Le paramètre Event peut prendre les formes suivantes :

MQISDP_MSG_DELIVERED : Le message a été envoyé/distribué (conformément au paramètre Qos).

MQISDP_MSG_DISCARDED : Le message a été supprimé (échec de toutes les tentatives).

Le paramètre Msgid contient l'id du message concerné par l'événement. L'id du message est renvoyé lors de l'utilisation de la fonction mqisdp_subscribe.

Le paramètre SubscriptionArray est un tableau d'éléments de la structure suivante :

```
Typedef struct
{
ascii* Topic,
u8 QoS
}
mqisdp_subscriptionParams_t;
```

Le tableau conserve des informations sur le niveau accordé de qualité de service pour chacune des rubriques souscrites.

* MsgHandler :

Programme de traitement de message par lequel le client reçoit des messages du courtier. Le programme de traitement utilise le type suivant :

```
Typedef bool (*mqisdp_msgHdlr_f) (
Ascii * Topic,
Bool Retain,
Bool Duplicate,
mqisdp_qos_e QoS,
Mqisdp_messageid msgid,
U16 PayloadLength,
U8 Payload [1]
);
```

Si le programme de traitement de message renvoie la valeur True, la bibliothèque supprime le paramètre Payload. Sinon, le client doit prendre en charge la gestion de la charge utile.

Le paramètre mqisdp_messageid est défini comme suit :

```
Typedef u16 mqisdp_messageid;
```

Le paramètre mqisdp_qos_e est défini comme suit :

```
Typedef enum
{
MQISDP_QOS_0,
MQISDP_QOS_1,
MQISDP_QOS_2,
} mqisdp_qos_e
```

c - Valeurs de retour

MQISDP_OK : Succès

MQISDP_ERR_Q_FULL : File d'attente des messages à destination de l'extérieur trop petite/saturée.

MQISDP_ERR_ALREADY_CONNECTED : Déjà connecté à un courtier. Commencez par déconnecter.

MQISDP_ERR_DISCONNECT_PENDING

//Une déconnexion est en suspens. Attente de

la déconnexion.

MQISDP_ERR_PARAM_CLIENT_ID : Le paramètre client_id est illégal.

MQISDP_ERR_PARAM_CLIENT_ID_TOO_BIG : Le paramètre client_id est trop long (maximum 23 caractères).

MQISDP_ERR_PARAM_TOPIC : Le paramètre topic est illégal.

MQISDP_ERR_PARAM_TOPIC_TOO_BIG : Le paramètre topic est trop long (maximum 32 767 caractères).

MQISDP_ERR_PARAM_DATA_LENGTH : Le paramètre payload_length est illégal.

MQISDP_ERR_PARAM_DATA_TOO_BIG : Le paramètre payload est trop long (maximum 65 535 caractères, y compris les en-têtes).

MQISDP_ERR_PARAM_QOS : Le paramètre qos est incorrect.

MQISDP_PIN_NOT_ENTERED : Une erreur du à un code PIN manquant s'est produite.

3.6.5 Fonction mqisdp_disconnect

Cette fonction est utilisée pour fermer la connexion mqisdp. Si elle est appelée alors qu'une autre déconnexion est en suspens, une erreur est renvoyée.

a - Prototype

```
u8 mqisdp_disconnect (
mqisdp_messageid* MsgId,
bool ImmediateDisconnect
);
```

b - Paramètres

* Msgid :

En retour, conserve l'id attribué au message de déconnexion.

* ImmediateDisconnect :

Cet indicateur détermine si la déconnexion doit être immédiatement forcée (si l'indicateur a la valeur True (vrai)) ou si la déconnexion peut attendre que tous

les messages actuellement en file d'attente soient distribués (ou supprimés). Si la déconnexion peut attendre, la bibliothèque n'accepte pas de nouveaux messages du courtier.

c - Valeurs de retour

MQISDP_OK : Succès.

MQISDP_ERR_DISCONNECT_PENDING : Une déconnexion est déjà en suspens.

3.6.6 Fonction mqisdp_publish

Cette fonction est utilisée pour publier un message MQIsdp au courtier de message. Cette opération ne peut être effectuée que si une connexion est déjà active.

a - Prototype

```
u8 mqisdp_publish (
mqisdp_messageId* MsgId,
ascii * Topic,
mqisdp_qos_e Qos,
bool Retain
u18 PayloadLength,
u8 Payload [1]
)
```

b - Paramètres

* Msgid :

En retour, conserve l'id attribué au message de déconnexion.

c - Valeurs de retour

MQISDP_OK : Succès.

MQISDP_ERR_CONNECTION_INVALID : La connexion au courtier n'est pas prête.

MQISDP_ER_Q_FULL : La file d'attente des messages à destination de l'extérieur est trop petite/saturée.

MQISDP_ERR_DISCONNECT_PENDING : Une déconnexion est en

suspens. Impossible d'accepter des messages.

MQISDP_ERR_PARAM_TOPIC : Le paramètre topic est illégal.

MQISDP_ERR_PARAM_TOPIC_TOO_BIG : Le paramètre topic est trop long (maximum 32 767 caractères).

MQISDP_ERR_PARAM_DATA_LENGTH : Le paramètre payload_length est illégal.

MQISDP_ERR_PARAM_DATA_TOO_BIG : Le paramètre payload est trop long (maximum 65 535 caractères, y compris les en-têtes).

MQISDP_ERR_PARAM_QOS : Le paramètre qos est incorrect.

3.6.7 - Fonction mqisdp_subscribe

Cette fonction est utilisée pour envoyer un message d'abonnement au courtier de message. Cette opération ne peut être effectuée que si une connexion est déjà active.

a - Prototype

```
u8 mqisdp_subscribe
{
mqisdp_messageId* MsgId,
u16 NoOfSubscriptions,
mqisdp_subscriptionParams_t*
SubscriptionArray
}
```

b - Paramètres

* Msgid :

En retour, conserve l'id attribué au message de déconnexion.

* NoOfSubscriptions :

Nombre d'éléments du paramètre SubscriptionArray.

* SubscriptionArray :

Tableau d'éléments de la structure suivante :

* Typedef struct

{

ascii * Topic,

u8 QoS

}

mqisdp_subscriptionParams_t;

c - Valeurs de retour

MQISDP_OK : Succès.

MQISDP_ERR_CONNECTION_INVALID : La connexion au courtier n'est pas prête.

MQISDP_ERR_Q_FULL : La file d'attente des messages à destination de l'extérieur est trop petite/saturée.

MQISDP_ERR_DISCONNECT_PENDING : //Une déconnexion est en suspens. Impossible d'accepter des messages.

MQISDP_ERR_PARAM_TOPIC : Le paramètre topic est illégal.

MQISDP_ERR_PARAM_TOPIC_TOO_BIG : Le paramètre topic est trop long (maximum 32 767 caractères).

MQISDP_ERR_PARAM_QOS : Le paramètre qos est incorrect.

3.6.8 - Fonction mqisdp_unsubscribe

Cette fonction est utilisée pour publier un message de désabonnement au courtier de message. Cette opération ne peut être effectuée que si une connexion est déjà active.

a - Prototype

```
u8 mqisdp_unsubscribe
{
mqisdp_messageId* MsgId,
u16 NoOfUnsubscriptions,
ascii * UnsubscriptionTopicAray
}
```

b - Paramètres

* Msgid :

En retour, conserve l'id attribué au message de déconnexion.

* NoOfUnsubscriptions :

Nombre d'éléments du paramètre UnsubscriptionArray.

* UnsubscriptionArray :

Tableau des rubriques concernées par le désabonnement.

c - Valeurs de retour

MQISDP_OK : Succès.

MQISDP_ERR_CONNECTION_INVALID : La connexion au courtier n'est pas prête.

MQISDP_ERR_Q_FULL : La file d'attente des messages à destination de l'extérieur est trop petite/saturée.

MQISDP_ERR_DISCONNECT_PENDING : //Une déconnexion est en suspens. Impossible d'accepter des messages.

MQISDP_ERR_PARAM_TOPIC : Le paramètre topic est illégal.

MQISDP_ERR_PARAM_TOPIC_TOO_BIG : Le paramètre topic est trop long (maximum 32 767 caractères).

3.6.9 - Fonction mqisdp_getConStatus

Cette fonction est utilisée pour demander le statut de la connexion.

a - Prototype

```
u8 mqisdp_getConStatus ();
```

b - Paramètres

Aucun

c - Valeurs de retour

MQISDP_CONNECTING : Si une connexion est en cours d'établissement (pas encore prête à accepter des messages).

MQISDP_CONNECTED : Si une connexion est active (et prête à accepter

des messages).

MQISDP_DISCONNECTED : Aucune connection n'est active.

3.6.10 -Fonction mqisdp_getMsgStatus

Cette fonction est utilisée pour demander le statut d'un message donné.

a - Prototype

```
u8 mqisdp_getMsgStatus (mqisdp_messageid* MsgId);
```

b - Paramètres

L'identifiant (id) du message.

c - Valeurs de retour

MQISDP_IN_QUEUE : Le message est dans la file d'attente à destination de l'extérieur (n'a pas encore été envoyé).

MQISDP_SENDING : Le message est maintenant en cours d'envoi (par TCP/IP).

MQISDP_IN_PROGRESS : Le message a été envoyé par TCP/IP mais aucun accusé de réception n'a encore été reçu.

MQISDP_NO_SUCH_MSG : Le message n'existe pas (plus). Soit il n'a jamais été envoyé, soit il a été envoyé et reçu (accusé de réception de la part du courtier).

3.6.11 - Fonction mqisdp_setQueueSize

Cette fonction est utilisée pour définir la taille de la file d'attente. La taille de la file d'attente peut être modifiée à tout moment, mais si la nouvelle taille est trop petite pour contenir les messages actuels, une erreur est renvoyée.

a - Prototype

```
u8 mqisdp_setQueueSize (u16 QueueSize);
```

b - Paramètres

QueueSize : Taille de la file d'attente en octets.

c - Valeurs de retour

MQISDP_OK : Succès.

MQISDP_ERR_Q_FULL : La taille de la file d'attente ne peut pas être définie. Le paramètre est sans doute trop petit pour les messages actuellement en file d'attente.

3.6.12 Fonction mqisdp_getQueueSize

Cette fonction est utilisée pour connaître la taille actuelle de la file d'attente.

a - Prototype

```
u16 mqisdp_get QueueSize ();
```

b - Paramètres

Aucun.

c - Valeurs de retour

La taille actuelle de la file d'attente est renvoyée (en octets).

3.6.13 - Fonction mqisdp_getAvailableSize

Cette fonction est utilisée pour connaître la quantité d'espace disponible dans la file d'attente.

a - Prototype

```
u16 mqisdp_getAvailableSize ();
```

b - Paramètres

Aucun.

c -Valeurs de retour

L'espace actuellement disponible dans la file d'attente est renvoyé (en octets).

3.7 Exemple de mise en œuvre

La figure 3 présente un exemple d'envoi de message, à l'aide des fonction API de l'invention.

Les étapes sont les suivantes :

- initialisation 31 : fonction « mqisdp_init() ;
- paramétrage 32 de la dimension de la file d'attente : fonction mqisdp_setqueuesize () ;

- connexion TCP/IP 33 (ouverture d'une session GPRS) ;
- si la connexion n'est pas bonne (34) : message d'erreur 35 ;
- sinon : connexion 36 au « courtier » : fonction mqisdp_connect () ;
- si la connexion n'est pas bonne (37) : message d'erreur 35 ;
- sinon : envoi du message 38 : fonction mqisdp_publish() ;
- si déconnexion 39, alors fin 310 ;
- sinon message d'erreur 35.

REVENDICATIONS

1. Système de contrôle d'équipements à distance, permettant l'interconnexion entre au moins un serveur et au moins un équipement distant selon le protocole MQIsdp,
caractérisé en ce qu'il associe à au moins un desdits équipements distants des moyens de radiocommunication capable de traiter en interne un protocole de communication mettant en oeuvre des fonctions source de type API disponibles dans une plateforme logicielle (Open AT) permettant d'embarquer au moins une application,
et en ce que lesdits moyens de radiocommunication sont dotés d'un jeu de fonctions (API) spécifiques permettant d'échanger des données avec au moins un serveur mettant en œuvre ledit protocole MQIsdp,
de façon à permettre une interconnexion entre le ou lesdits serveurs et le ou lesdits équipements distants via lesdits moyens de radiocommunication, ces derniers gérant également au moins une application entre le ou lesdits serveurs et le ou lesdits équipements distants.

2. Système de contrôle d'équipements à distance selon la revendication 1, caractérisé en ce que lesdits moyens de radiocommunication comprennent un module de radiocommunication, regroupant sur un même substrat l'ensemble des moyens de traitement de données radio-fréquence et bande de base, ainsi que les moyens de gestion desdites fonctions (API) et la ou lesdites applications.

3. Système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 et 2, caractérisé en ce que lesdits moyens de radiocommunication intègrent ledit protocole MQIsdp sous la forme d'une librairie, définissant ledit jeu de fonctions (API) spécifiques.

4. Système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 à 3, caractérisé en ce que, au moins dans un premier mode, lesdits moyens de radiocommunication gèrent uniquement la signalisation d'un

échange de données, lesdites données étant transférées directement d'un équipement distant vers un serveur, ou inversement.

5. Système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 à 4, caractérisé en ce que, au moins dans un second mode, lesdits moyens de radiocommunication gèrent la signalisation d'un échange de données et le transfert desdites données, ces dernières étant temporairement stockées dans au moins une mémoire tampon.

6. Système de contrôle d'équipements à distance selon la revendication 5, caractérisé en ce que la taille de la ou desdites mémoires tampon est paramétrable.

7. Système de contrôle d'équipements à distance selon les revendications 4 et 6, caractérisé en ce qu'il fonctionne dans ledit premier mode lorsque la taille de la ou desdites mémoires tampon vaut 0, et dans ledit second mode sinon.

8. Système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 à 7, caractérisé en ce que ledit jeu de fonctions API spécifiques comprend des fonctions permettant :

- la connexion à un desdits serveurs ;
- l'envoi de messages ;
- la réception de messages ;
- configuration d'au moins un paramètre.

9. Système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 à 8, caractérisé en ce qu'au moins certaines desdites fonctions (API) spécifiques sont organisées de façon à pouvoir assurer au moins deux opérations et/ou agir sur au moins deux aspects distincts, en fonction d'un paramétrage prédéfini.

10. Système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 à 9, caractérisé en ce que ledit jeu de fonctions (API) comprend uniquement 12 fonctions.

11. Système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 à 10, caractérisé en ce que ledit jeu de fonctions (API) spécifiques comprend une fonction d'initialisation (mqisdp_init) rétablissant des

paramètres par défaut, et devant être appelée au moins une fois avant l'utilisation d'autres fonctions (API).

12. Système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 à 11, caractérisé en ce que ledit jeu de fonctions (API) spécifiques comprend une fonction (mqisdp_resume) appelée lorsque une liaison IP a été établie.

13. Système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 à 12, caractérisé en ce qu'il comprend une fonction d'établissement d'une connexion avec un desdits serveurs (mqisdp_connect), permettant de définir des paramètres de ladite connexion, et une fonction de déconnexion (mqisdp_disconnect) de ladite connexion.

14. Système de contrôle d'équipements à distance selon la revendication 13, caractérisé en ce que ladite fonction d'établissement d'une connexion permet de sélectionner un mode de transmission parmi au moins deux (GSM et GPRS).

15. Système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 à 14, caractérisé en ce qu'il comprend une fonction (mqisdp_publish) pour l'envoi d'un message vers un desdits serveurs.

16. Système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 à 15, caractérisé en ce qu'il comprend une fonction d'abonnement à un desdits serveurs (mqisdp_subscribe), et une fonction de désabonnement (mqisdp_unsubscribe) audit serveur.

17. Système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 à 16, caractérisé en ce qu'il comprend au moins une fonction de demande d'information sur au moins un aspect d'une communication en cours.

18. Système de contrôle d'équipements à distance selon la revendication 17, caractérisé en ce qu'il comprend au moins une des fonctions appartenant au groupe comprenant :

- une fonction de demande du statut d'une connexion (mqisdp_getConStatus) ;
- une fonction de demande du statut d'un message donné

(mqisdp_getMsgStatus) ;

- une fonction de demande de la taille actuelle d'une file d'attente (mqisdp_getQueueSize) ;
- une fonction de demande de l'espace disponible dans une file d'attente (mqisdp_getAvailableSize).

19. Système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 à 18, caractérisé en ce qu'il comprend une fonction de définition de la taille d'une file d'attente (mqisdp_setQueueSize).

20. Procédé de contrôle d'équipements à distance, permettant l'interconnexion entre au moins un serveur et au moins un équipement distant selon le protocole MQIsdp,

caractérisé en ce qu'il associe à au moins un desdits équipements distants des moyens de radiocommunication capable de traiter en interne un protocole de communication mettant en oeuvre des fonctions source de type API disponibles dans une plateforme logicielle (Open AT) permettant d'embarquer au moins une application,

et en ce qu'il met en œuvre, dans lesdits moyens de radiocommunication, un jeu de fonctions API spécifiques permettant d'échanger des données avec au moins un serveur mettant en œuvre ledit protocole MQIsdp,

de façon à permettre une interconnexion entre le ou lesdits serveurs et le ou lesdits équipements distants via lesdits moyens de radiocommunication, ces derniers gérant également au moins une application entre le ou lesdits serveurs et le ou lesdits équipements distants.

21. Dispositif de radiocommunication caractérisé en ce qu'il comprend des moyens de radiocommunication mis en œuvre dans un système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 à 19.

22. Module de radiocommunication caractérisé en ce qu'il comprend des moyens de radiocommunication mis en œuvre dans un système de contrôle d'équipements à distance selon l'une quelconque des revendications 1 à 19.

23. Jeu de fonctions (API) mis en oeuvre dans un système de contrôle

d'équipements à distance, caractérisé en ce qu'il permet d'échanger des données avec au moins un serveur mettant en œuvre ledit protocole MQIsdp.

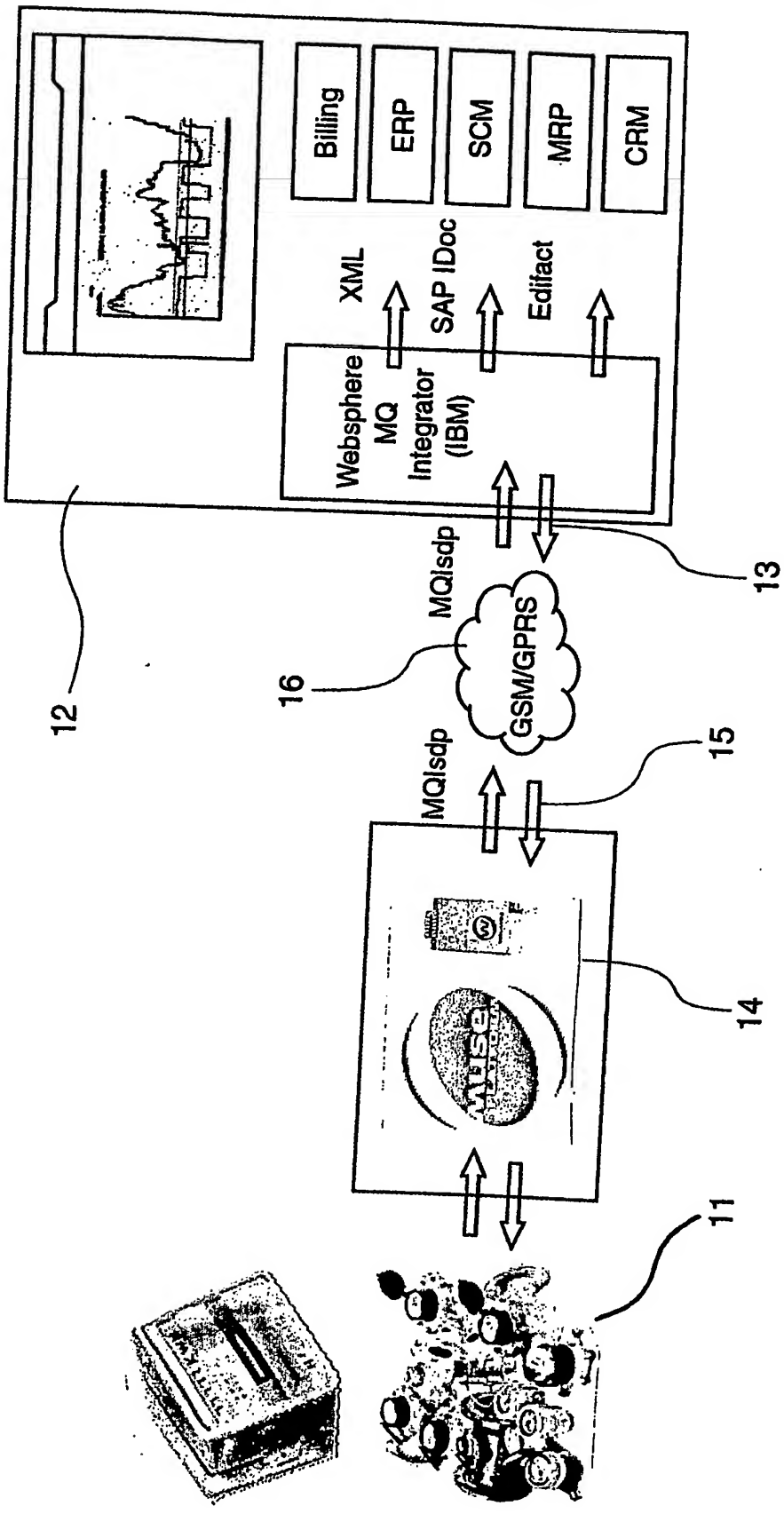
Billing
ERP
SCM
MRP
CRM
XML
SAP IDoc
Edifact
Websphere MQ Integrator (IBM)
MQIsdp
GSM/GPRS
MQIsdp
12
13
16
15
14
11

Fig. 1

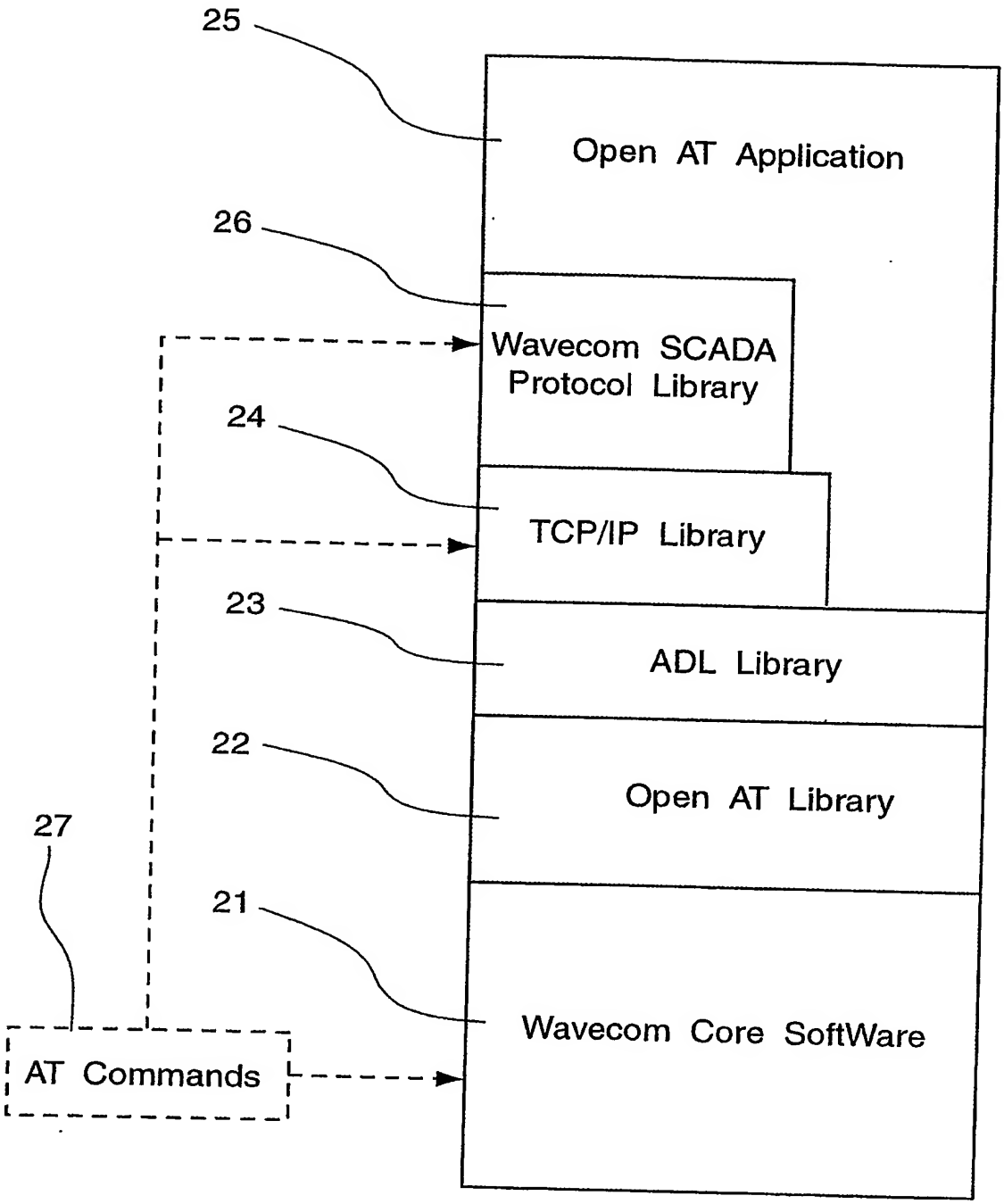
25
Open AT Application
26
Wavecom SCADA Protocol Library
24
TCP/IP Library
23
ADL Library
22
Open AT Library
27
21
Wavecom Core SoftWare
AT Commands

Fig. 2

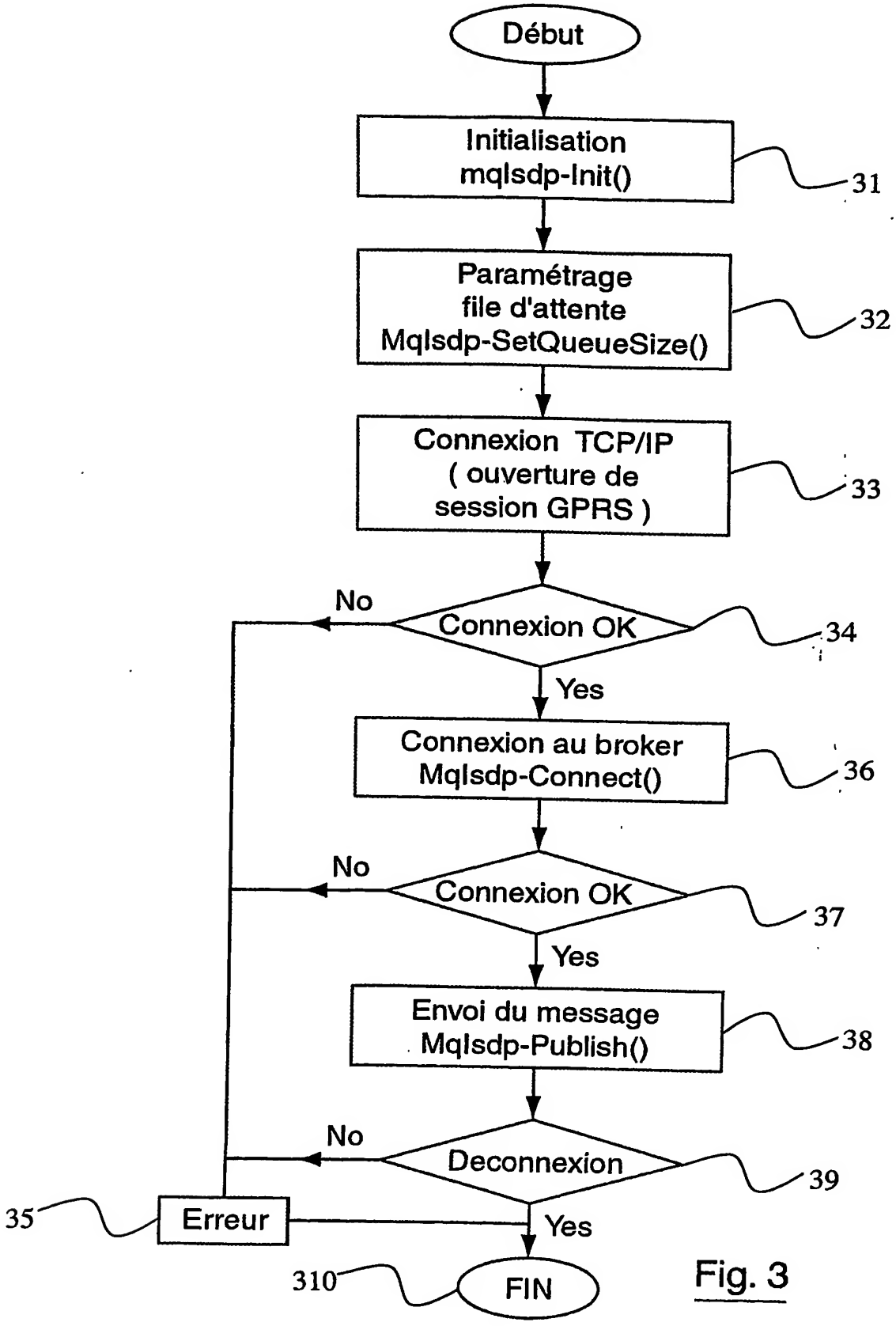
Début
Initialisation
mqIsdp-Init()
31
Paramétrage
file d'attente
MqIsdp-SetQueueSize()
32
Connexion TCP/IP
(ouverture de
session GPRS)
33
No
Connexion OK
34
Yes
Connexion au broker
MqIsdp-Connect()
36
No
Connexion OK
37
Yes
Envoi du message
MqIsdp-Publish()
38
No
Deconnexion
39
35
Erreur
Yes
310
FIN

Fig. 3